Universidade Federal de Alagoas (UFAL)
Campus Arapiraca

Programação Orientada a Objetos (POO)

09 - Representação gráfica de classes (Diagrama de classes)

Alexandre de Andrade Barbosa

alexandre.barbosa@arapiraca.ufal.br

Objetivos

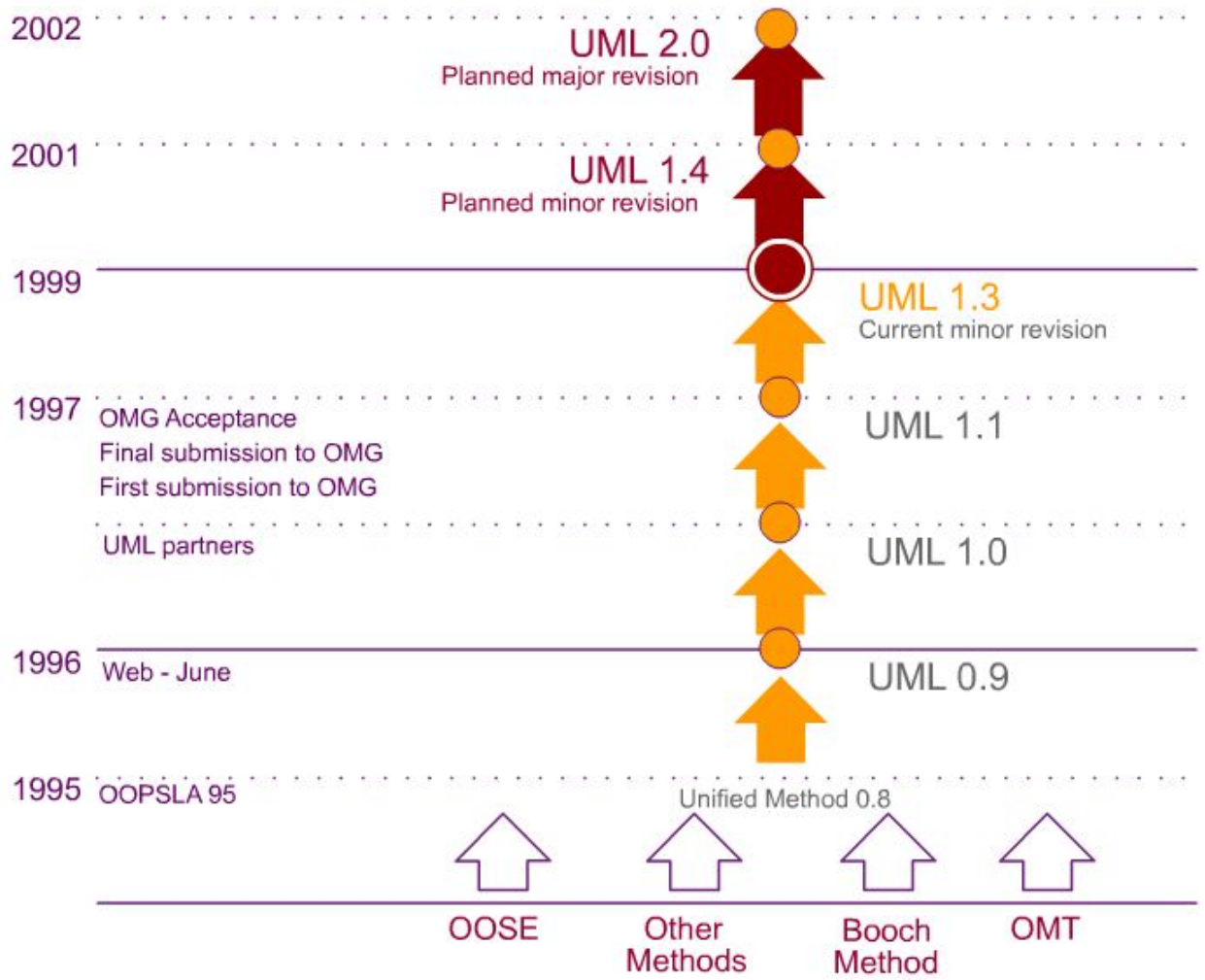
Objetivos

- Apresentar brevemente a linguagem UML
- Exibir os conceitos de diagrama de classes
- Apresentar traduções possíveis de um diagrama de classes para código

Modelagem
Introdução

- Modelos fornecem uma visão simplificada de um sistema.
- Modelos servem para:
  - visualizar;
  - descrever estrutura e/ou comportamento;
  - auxiliar em decisões;
  - documentar.
- Quanto mais diagramas criar melhor meu projeto será?
  - Não é necessário criar todos os modelos existentes, e de forma extremamente detalhada!!!

Unified Modelling Language (UML)

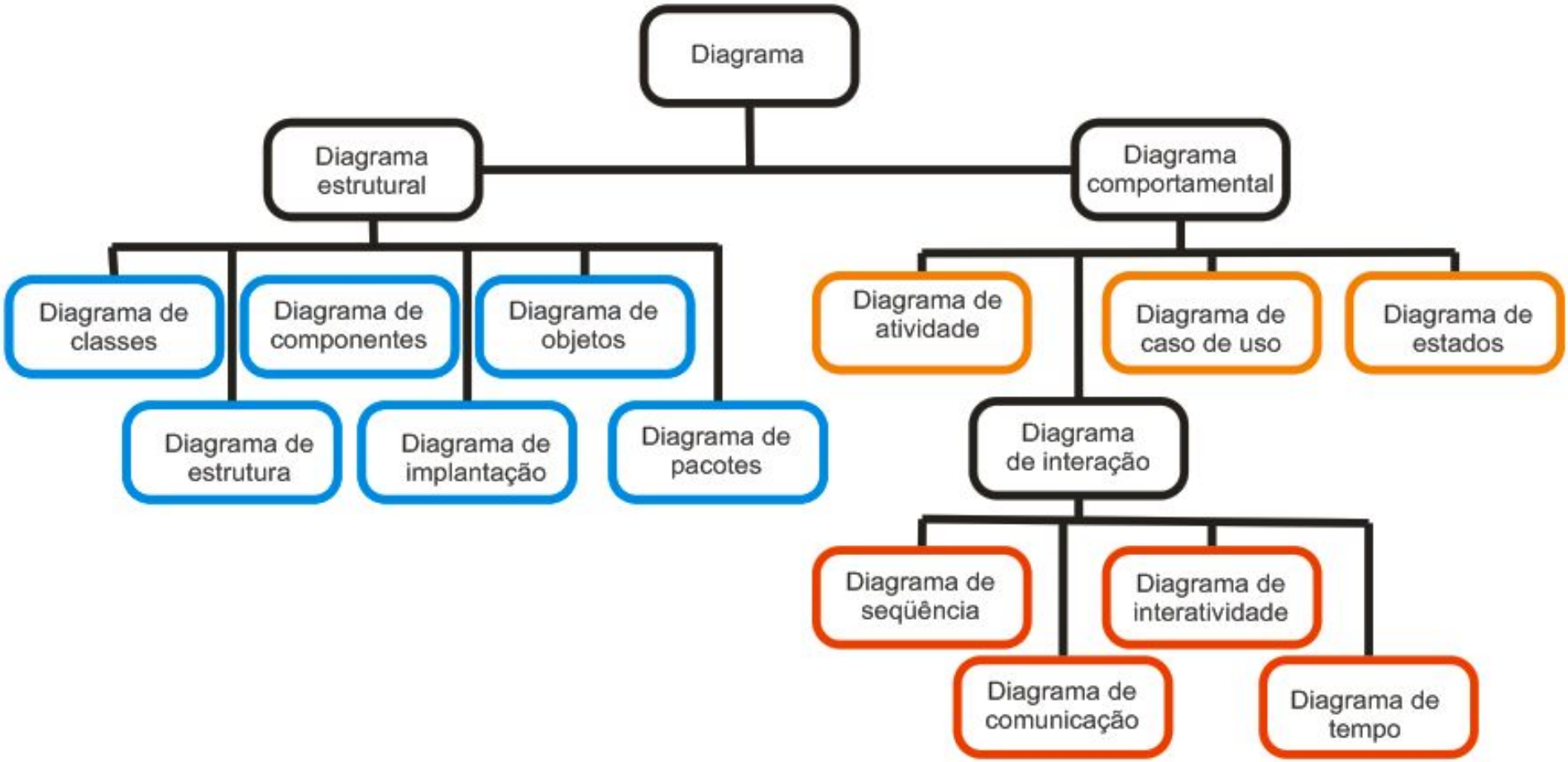


Unified Modelling Language (UML)

- UML é uma linguagem de modelagem, toda linguagem sofre mudanças ao longo do tempo.

UML 1.4	UML 2.0
Diagrama de classes	Diagrama de classes
Diagrama de objetos	Diagrama de objetos
Diagrama de componentes	Diagrama de componentes
Diagrama de implantação	Diagrama de implantação
-	Diagrama de pacotes
-	Diagrama de estrutura
Diagrama de caso de uso	Diagrama de caso de uso
Diagrama de estados	Diagrama de estados
Diagrama de atividade	Diagrama de atividade
Diagrama de seqüência	Diagrama de seqüência
-	Diagrama de interatividade
Diagrama de colaboração	Diagrama de comunicação
-	Diagrama de tempo

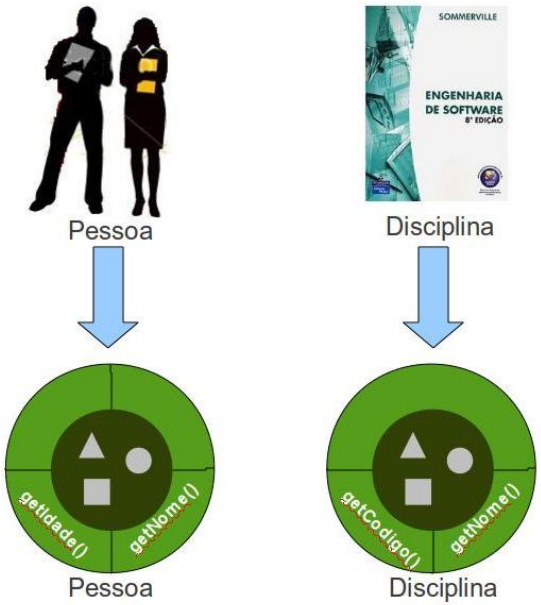
Unified Modelling Language (UML)



Ferramentas

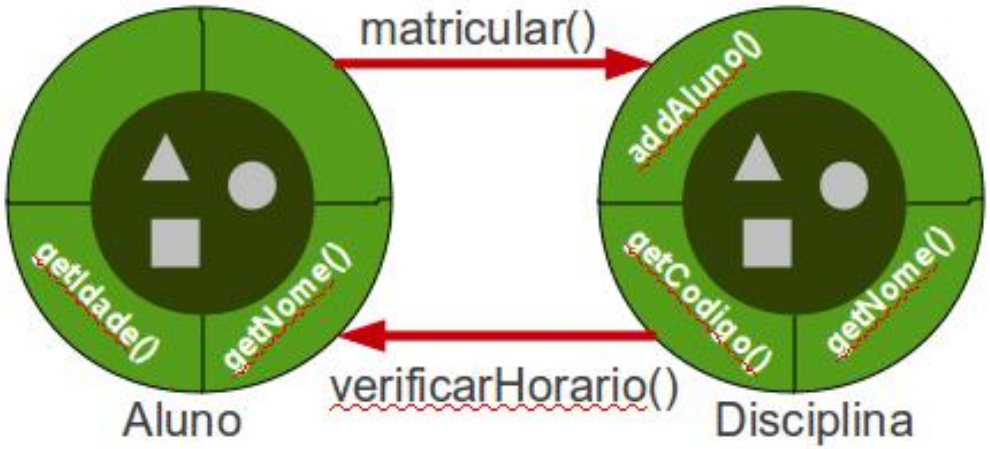
- Modelagem UML
  - Dia (Link)
  - Eclipse Modeling Tools (Link)
- Trello (Link)

Projeto de software
Classes e objetos



- Objetos são abstrações do mundo real ou entidades de sistema e gerenciam a si próprios
- Os objetos são independentes, eles englobam estados e informações de representação.

Projeto de software
Classes e objetos



- A funcionalidade do sistema é expressa em termos de serviços de objetos.
- Colaborações entre objetos ocorrem através de troca de mensagens

Projeto de software
Classes e objetos

"Um objeto é uma entidade que possui um estado e um conjunto de operações definidas para funcionar nesse estado. O estado é representado como um conjunto de atributos de objeto. As operações associadas ao objeto fornecem serviços a outros objetos (clientes) que solicitam estes serviços quando alguma computação é necessária."

Sommerville, 2007

Projeto de software
Classes e objetos

"Os objetos são criados de acordo com uma definição de classe de objeto. Uma definição de classe de objeto funciona tanto como uma especificação quanto como um template para criação de objetos. Essa definição inclui declarações de todos os atributos e operações que devem ser associadas a um objeto dessa classe."

Sommerville, 2007

Diagrama de classes (DC)

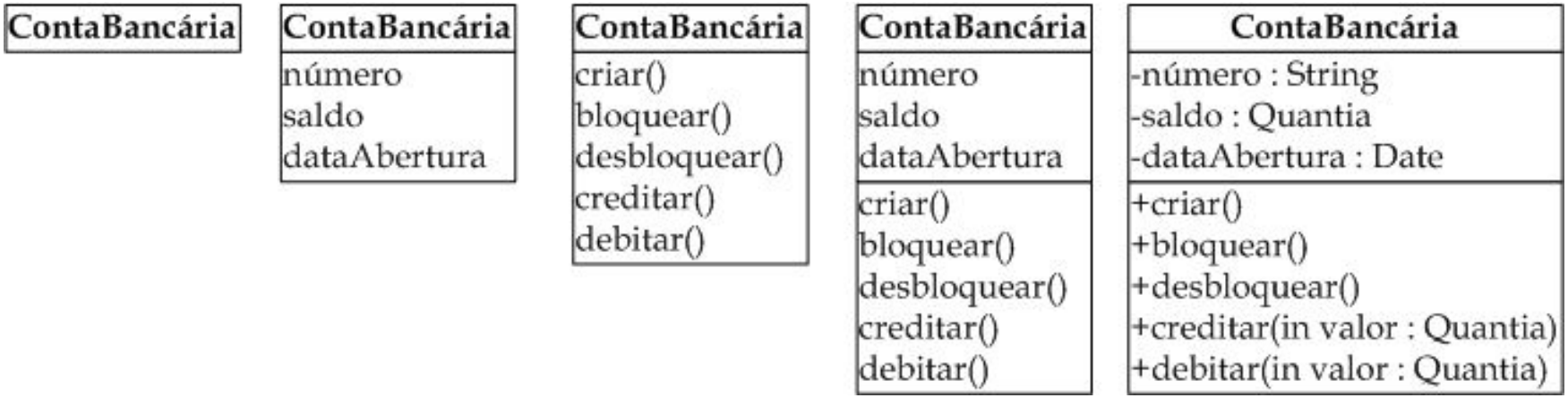
Classes, atributos e métodos

- Classe. Abstração de uma entidade conceitual ou real
- Uma classe define os comportamentos (métodos) e as características (atributos) da entidade modelada

Representações de classes em UML

Diagrama de classes (DC)

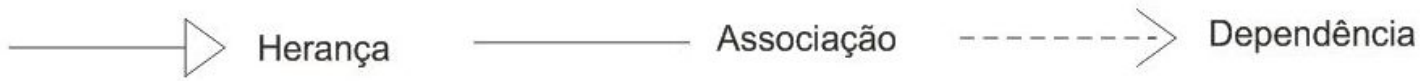
Classes, atributos e métodos



Representações de classes em UML

Projeto de software
Classes e objetos

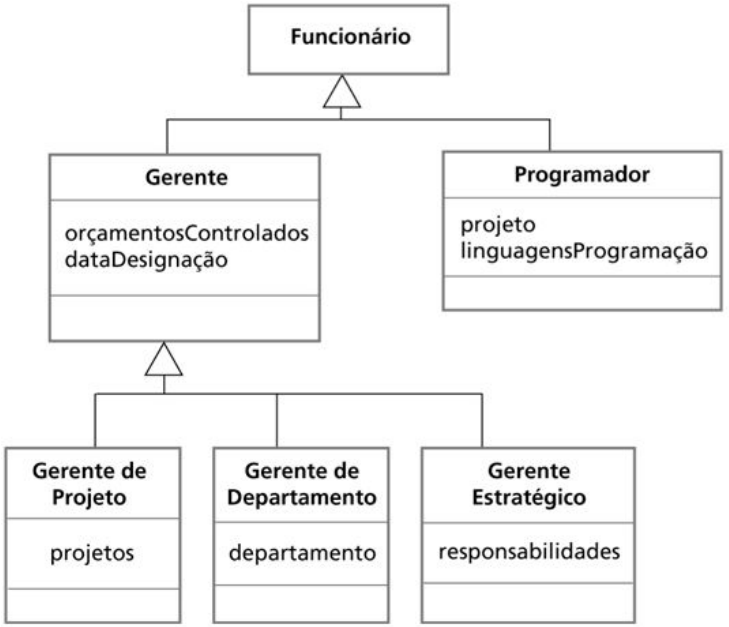
- Objetos são membros de classes que definem tipos de atributos e operações
- As classes sempre estão relacionadas a outras classes, pois seus objetos precisam se comunicar



Relacionamentos

Projeto de software
Classes e objetos

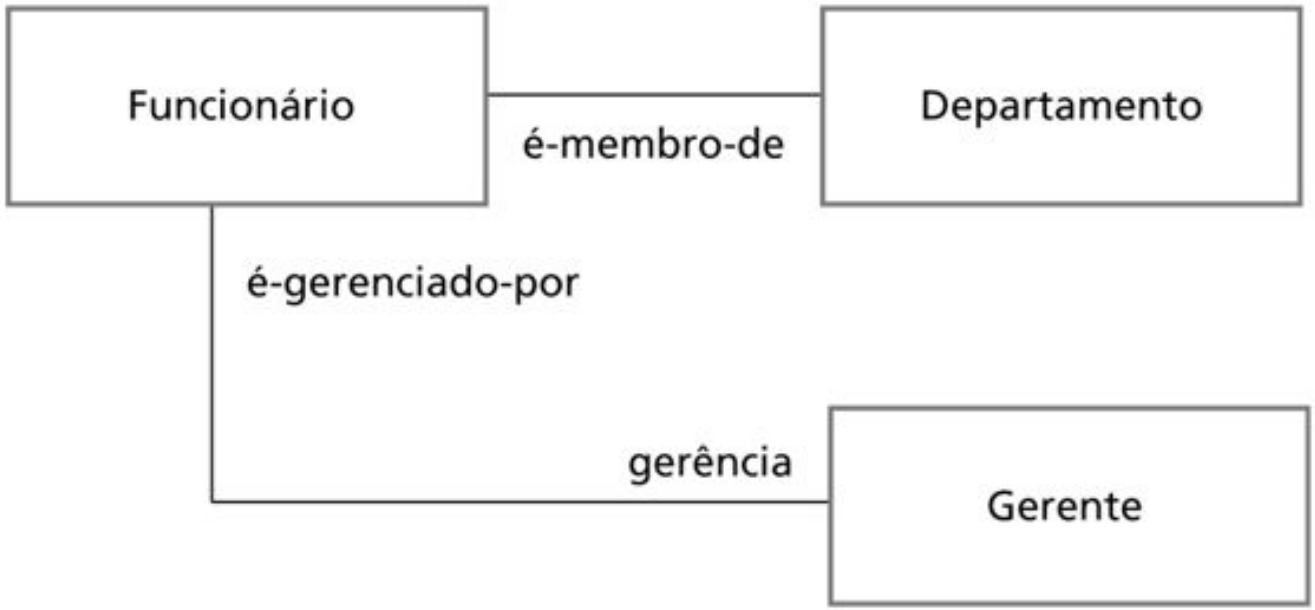
- As classes podem ser organizadas em uma hierarquia onde uma classe (uma classe-pai) é uma generalização de uma ou mais classes (classes-filho).
- Uma classe-filho herda os atributos e as operações da classe-pai e pode adicionar novos métodos ou atributos para si



Hierarquia de generalização

Projeto de software
Classes e objetos

- Objetos e classes de objeto participam de relacionamentos com outros objetos e classes de objeto
- Na UML, um relacionamento generalizado é indicado por uma associação

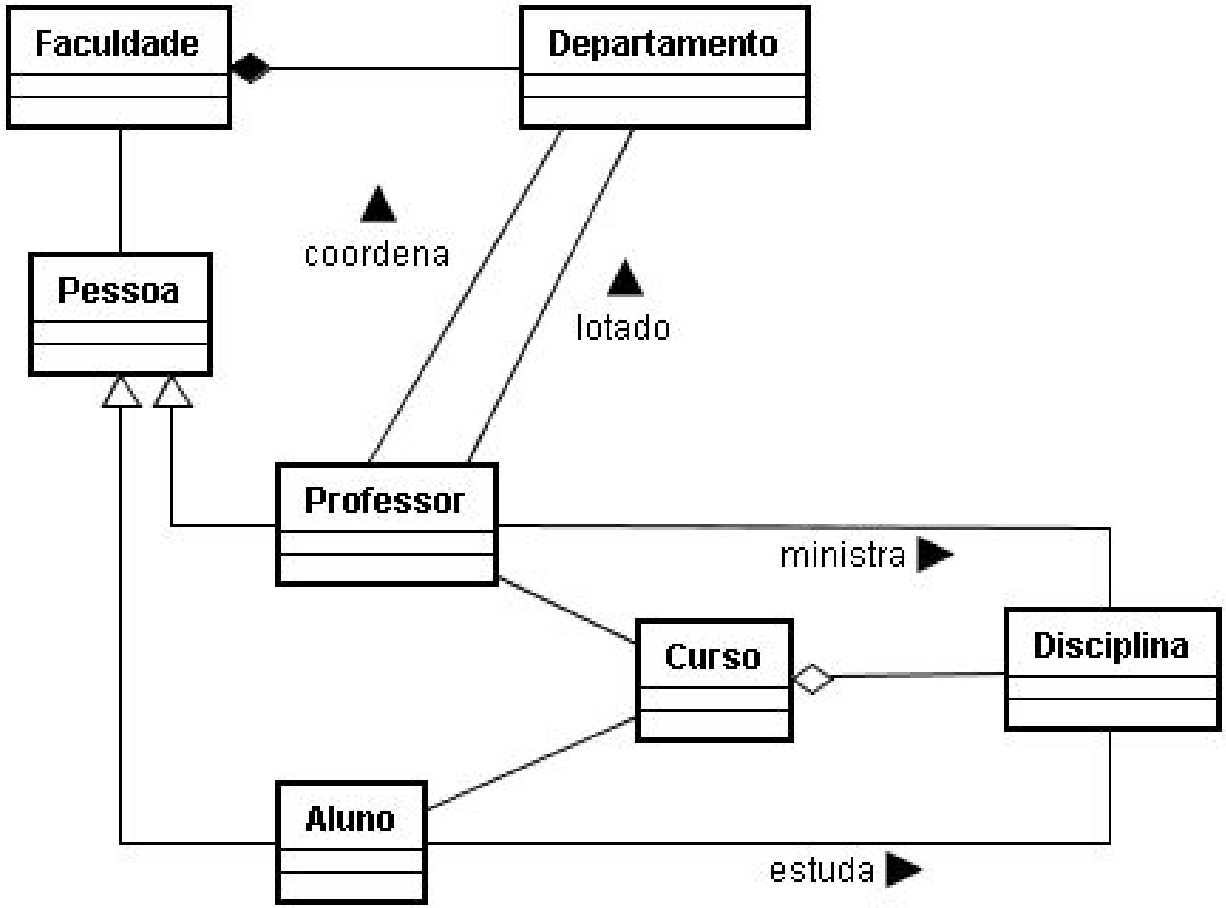


Associação

Projeto de software
Classes e objetos

- Outros elementos de um relacionamento:
  - multiplicidade
  - nome da associação
  - papéis
  - direção de leitura
  - visibilidade (modificadores de acesso)

Exemplos



Exemplo de um diagrama de classes

De UML para código

Classe: Nome

De UML para código

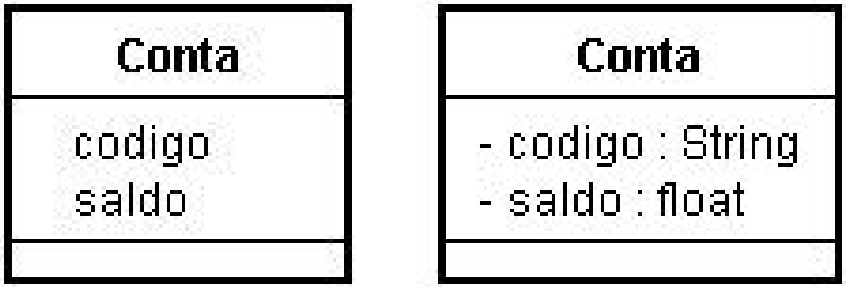
Classe: Nome


```
public class Conta {

}
```

```
public class ContaCorrente {

}
```

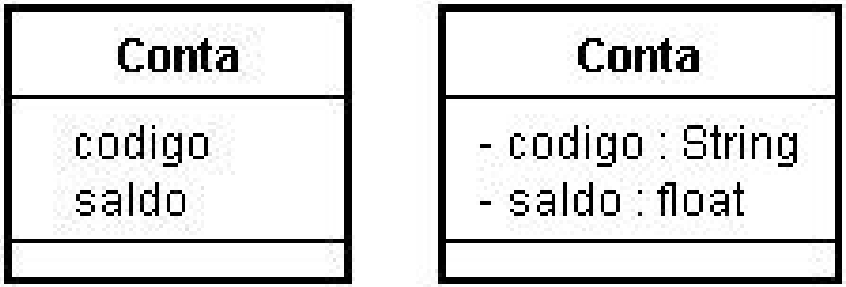
De UML para código

Classe: Atributo



De UML para código

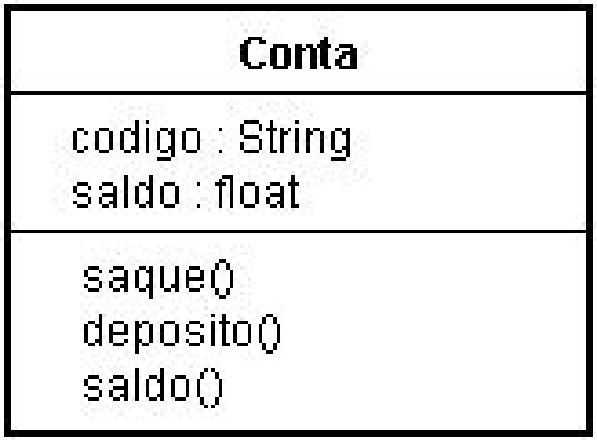
Classe: Atributo

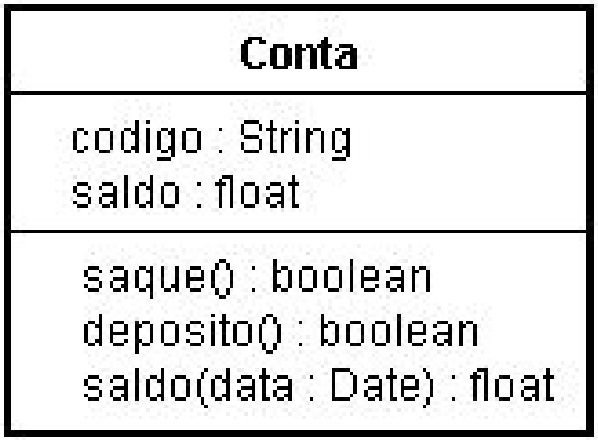



```
public class Conta {
    private String codigo;
    private float saldo;
}
```

De UML para código

Classe: Operação



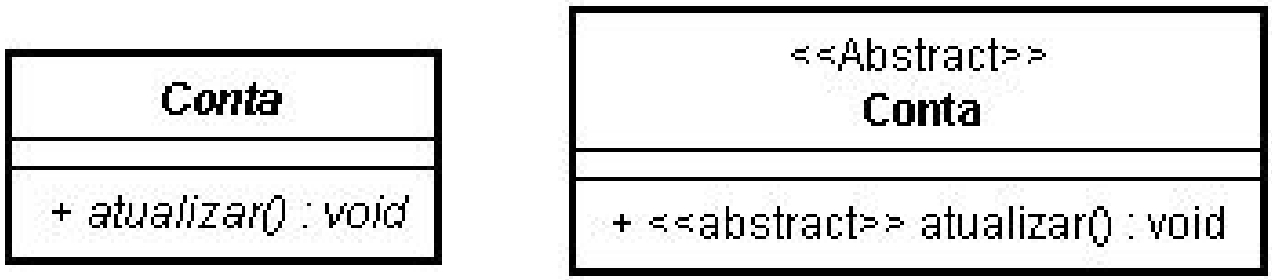


De UML para código

Classe: Operação

```
public class Conta {
    private String codigo;
    private float saldo;
    public boolean saque() {
        //...
    }
    public boolean deposito() {
        //...
    }
    public float saldo(Date data) {
        //...
    }
}
```

De UML para código
Classe abstrata



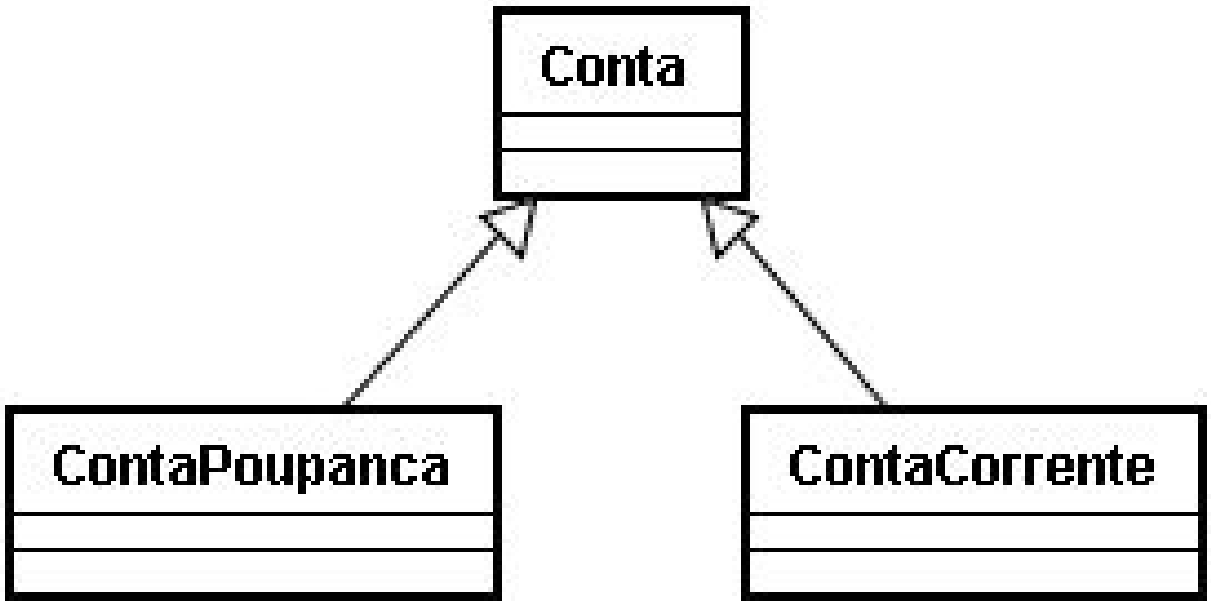

```
public abstract class Conta {
    public abstract void atualizar() {
        //...
    }
}
```

De UML para código
Interface


```
public interface RepositorioDeContas {

}
```

De UML para código

Relacionamentos: Herança



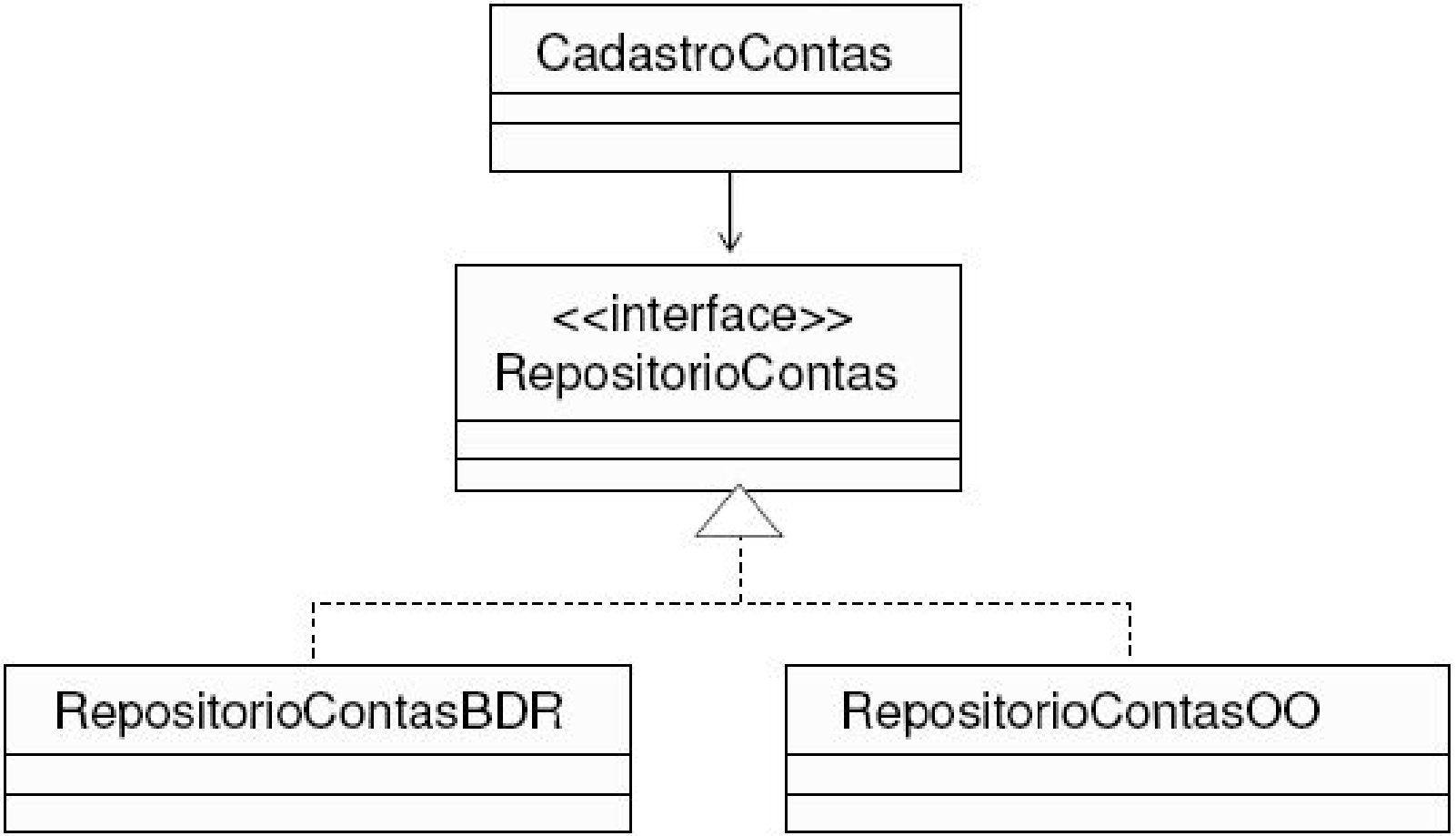
De UML para código

Relacionamentos: Herança

```
public class Conta {
    // ...
}
public class ContaCorrente extends Conta {
    // ...
}
public class ContaPoupanca extends Conta {
    // ...
}
```

De UML para código

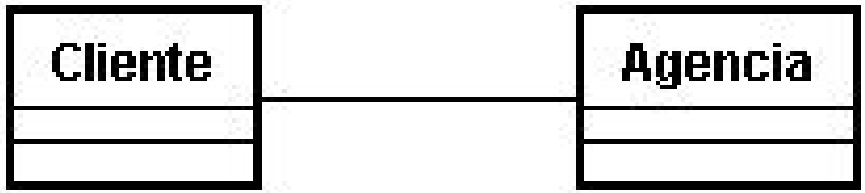
Relacionamentos: Realização



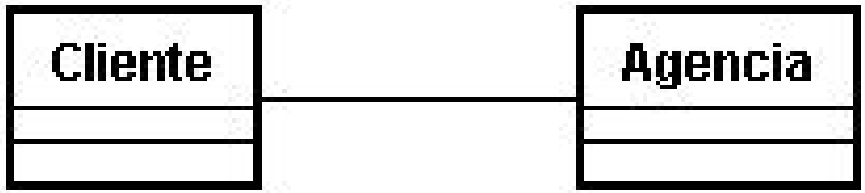
De UML para código
Relacionamentos: Realização

```
public interface RepositorioContas {
    // ...
}
public class RepositorioContasBDR implements RepositorioConta
    // ...
}
public class RepositorioContasOO implements RepositorioContas
    // ...
}
```

De UML para código
Relacionamentos: Associação



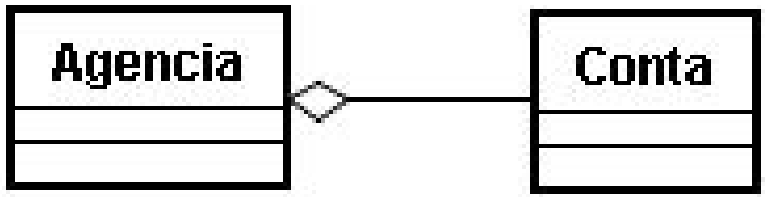
De UML para código
Relacionamentos: Associação




```
public class Cliente {
    private Agencia agencia;
    // ...
}

public class Agencia {
    private Cliente cliente;
    // ...
}
```

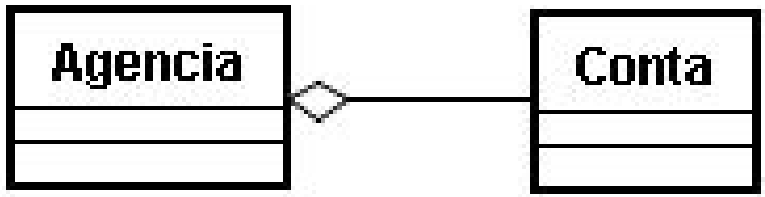
De UML para código

Relacionamento: Agregação



De UML para código

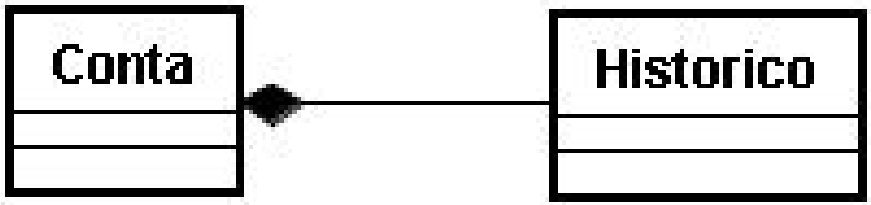
Relacionamento: Agregação




```
public class Agencia {
    private Conta conta;
    // ...
}
public class Conta {
    private Agencia agencia;
    // ...
}
```

- Distintos somente pelo significado associado.

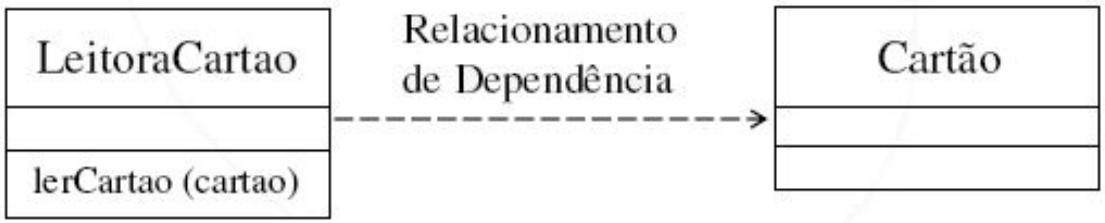
De UML para código
Relacionamento: Composição




```
public class Conta {
    private Historico historico;
    // ...  ---------------------> public void destroy() {
                                        historico.destroy();
}                                   }
public class Historico {
    private Conta conta;
    // ...
}
```

- Distintos somente pelo significado associado.

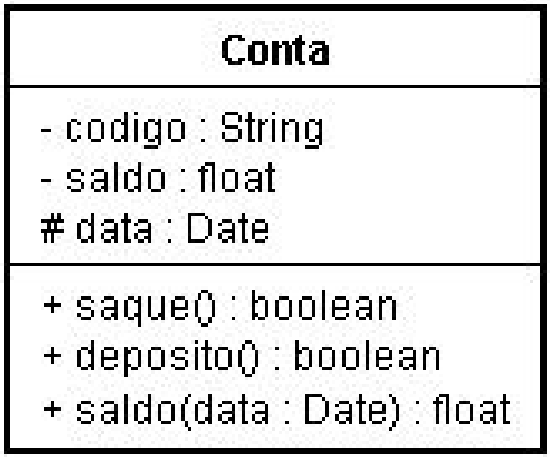
De UML para código
Relacionamentos: Dependência




```
public class LeitoraDeCartao {
    public boolean lerCartao(Cartao cartao) {
        // ...
    }
}
public class Cartao {
    // ...
}
```

De UML para código

Relacionamento: Visibilidade



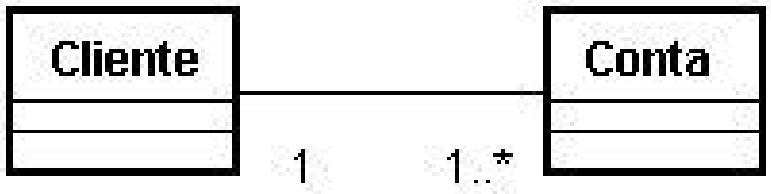
De UML para código

Relacionamento: Visibilidade

```
public class Conta {
    private String codigo;
    private float saldo;
    protected Date data;
    public boolean saque() {
        // ...
    }
    public boolean deposito() {
        // ...
    }
    public boolean saldo(Date data) {
       // ...
    }
}
```

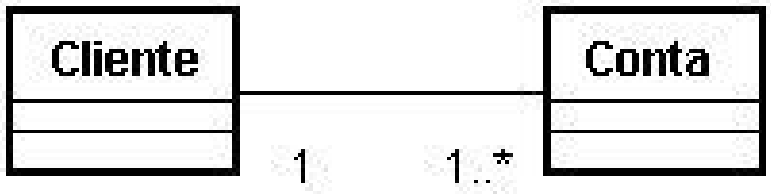
De UML para código

Classe: Multiplicidade



De UML para código

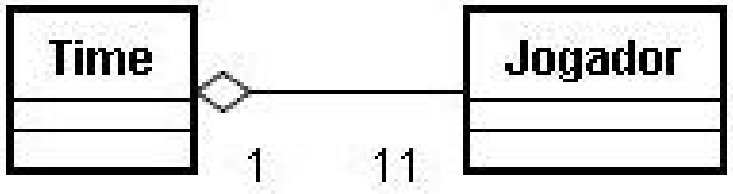
Classe: Multiplicidade




```
public class Cliente {
    private Vector<Conta> contas;
    // ...
}
public class Conta {
    private Cliente cliente;
    // ...
}
```

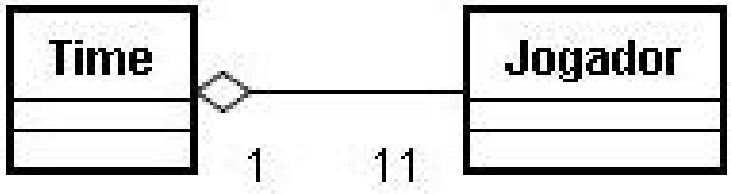
De UML para código

Classe: Multiplicidade



De UML para código

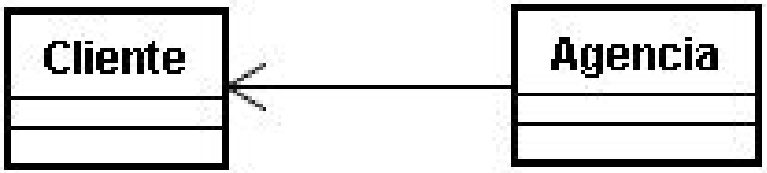
Classe: Multiplicidade




```
public class Time {
    private Jogador[] jogadores;
    public Time() {
        jogadores = new Jogador[11];
    }
    // ...
}
public class Jogador {
    private Time time;
    // ...
}
```

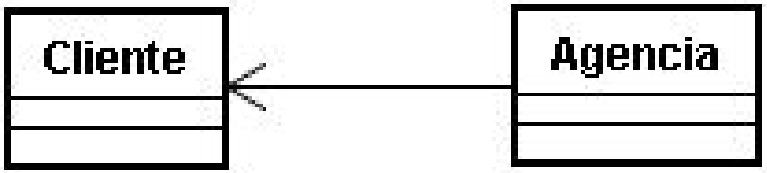
De UML para código

Classe: Navegabilidade



De UML para código

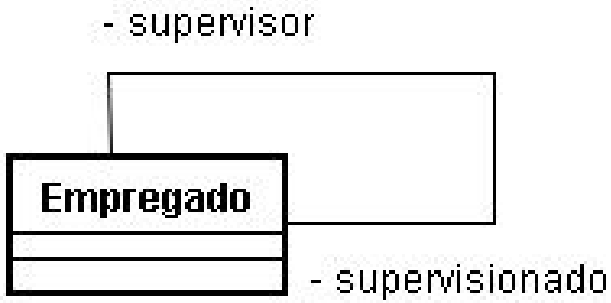
Classe: Navegabilidade




```
public class Cliente {
    // ...
}
public class Agencia {
    private Cliente cliente;
    // ...
}
```

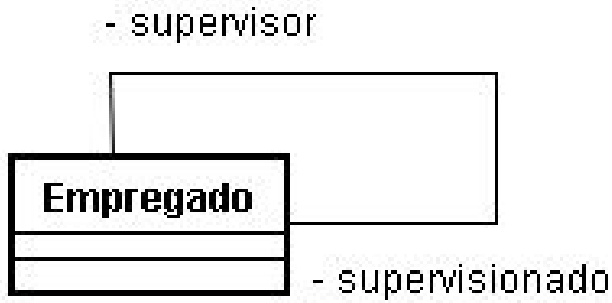
De UML para código

Classe: Pápeis



De UML para código

Classe: Pápeis



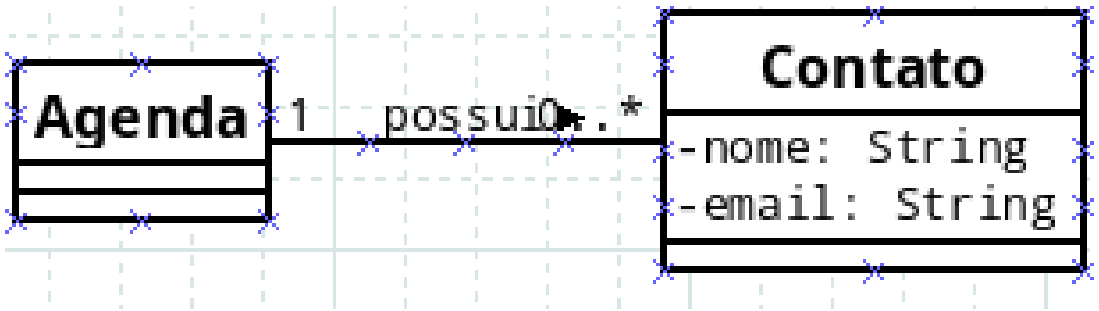

```
public class Pessoa {
    private Pessoa pessoa;
    // ...
}
public class Empregado {
    private Empregado supervisor;
    private Empregado supervisionado;
    // ...
}
```

Representação gráfica de classes (Diagrama de classes)

Exercícios

Exercício

1. Crie uma aplicação "Agenda de contatos" nessa aplicação um contato deve obedecer a descrição de classe exibida a seguir, você pode adicionar as operações que desejar. Deve ser possível adicionar um contato, listar os contatos cadastrados e remover um contato.

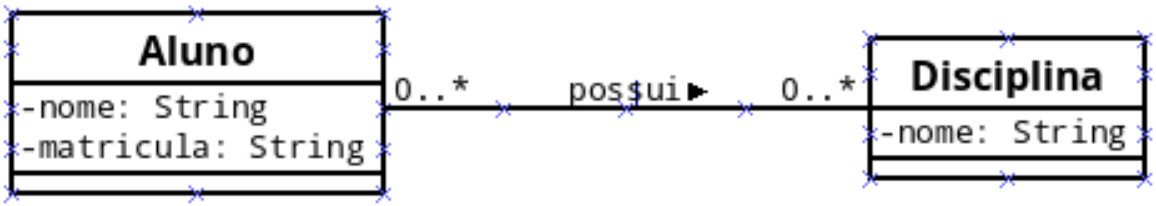


Representação gráfica de classes (Diagrama de classes)

Exercícios

Exercício

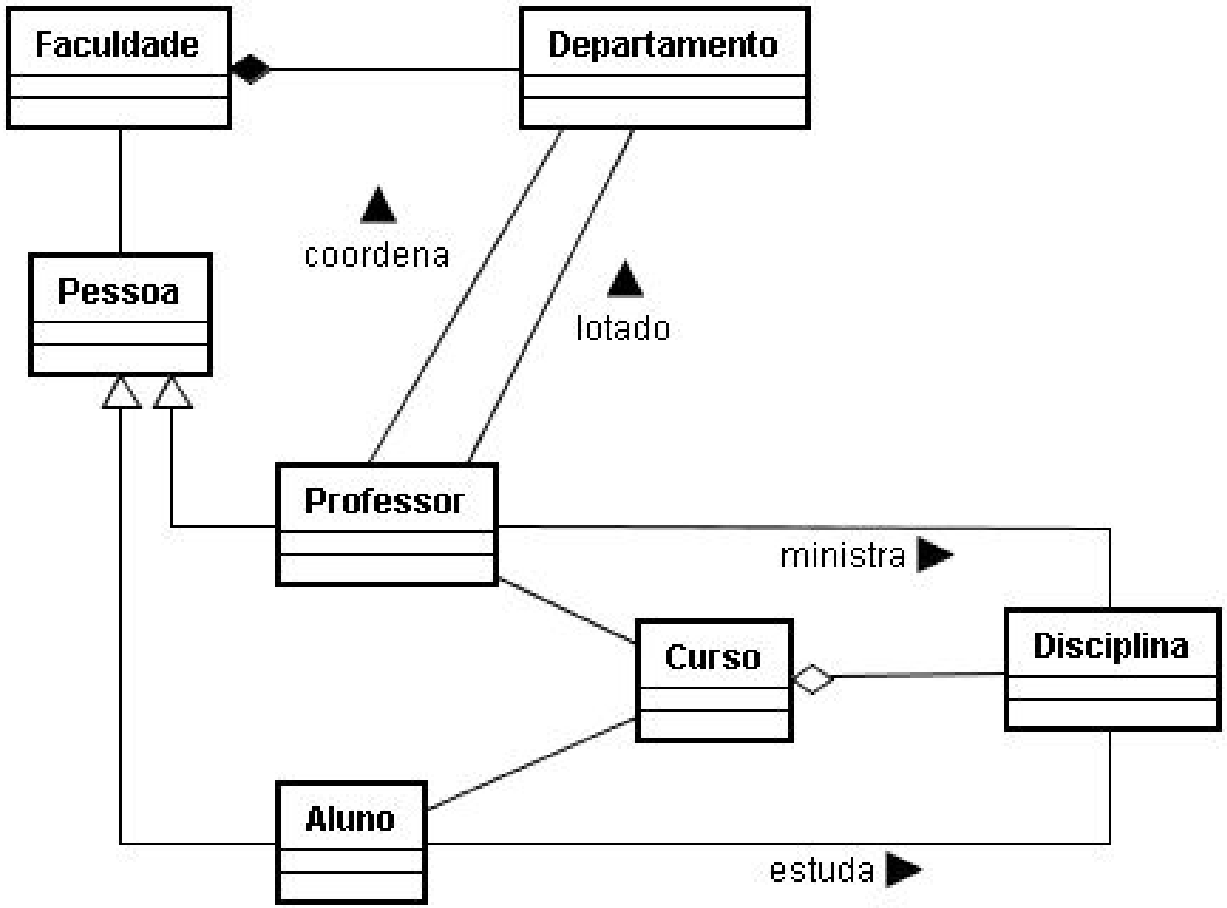
1. Crie uma aplicação onde existem entidades de "Aluno" e "Disciplina". Um "Aluno" deve obedecer a descrição de classe exibida a seguir. Uma "Disciplina" deve obedecer a descrição de classe exibida a seguir. Deve ser possível cadastrar alunos, listar alunos e remover alunos. Deve ser possível cadastrar disciplinas, listar disciplinas e remover disciplinas. Ainda deve ser possível realizar uma matricula, ou seja um aluno fará parte dos alunos de uma disciplina e uma disciplina será uma das disciplinas do aluno.



Exercício

Exercício

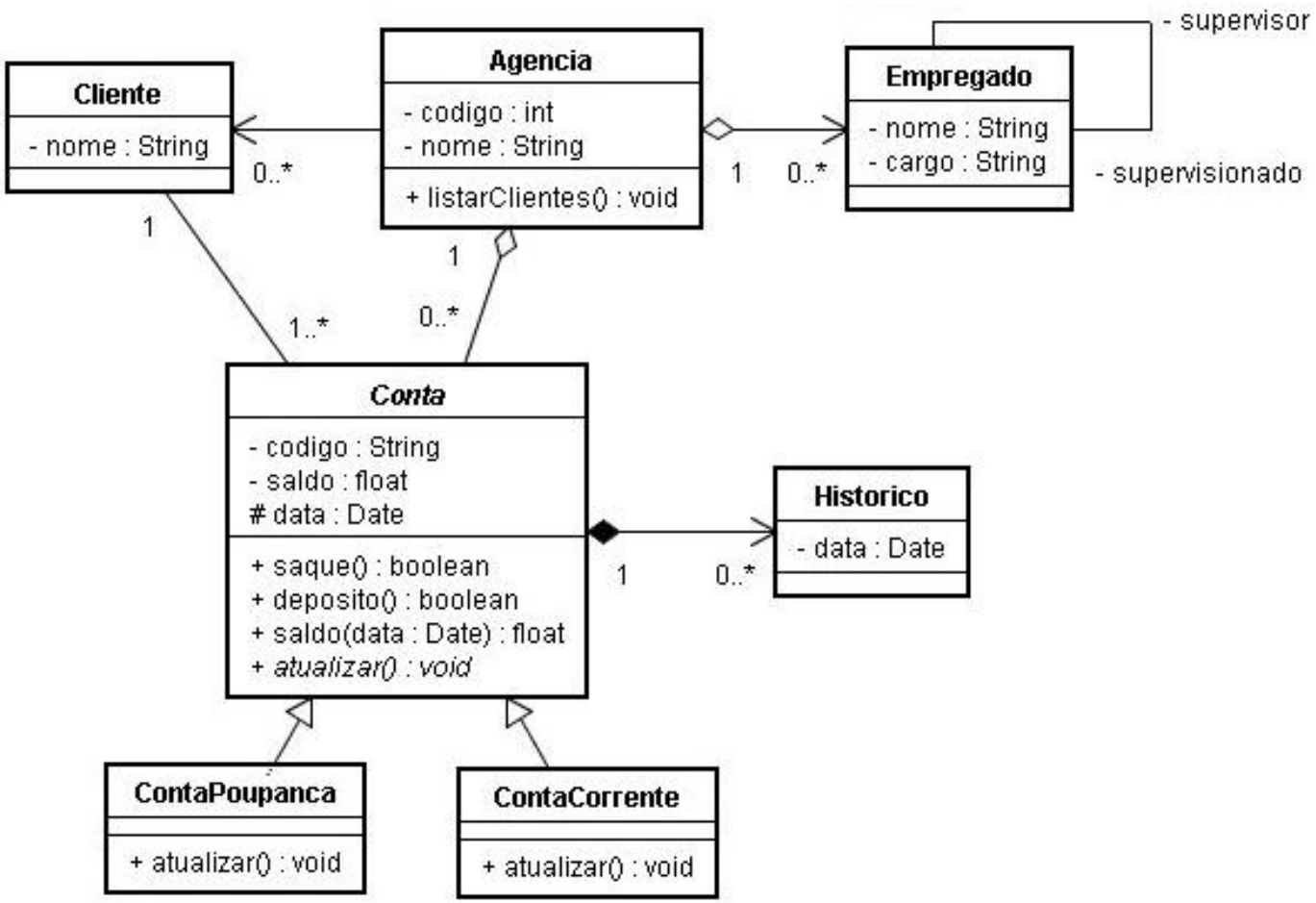
1. Traduza o diagrama de classes para código



Exercício

Exercício

1. Traduza o diagrama de classes para código



Resumo

- UML é um padrão para modelar sistemas de software
- Os diagramas UML podem ser classificados em: diagramas de estrutura, diagramas de comportamento e diagramas de interação
- O diagrama de classes é UM dos diversos diagramas existentes em UML

Leituras recomendadas

Gilleanes T. A. Guedes
UML - Uma abordagem prática, 2ª ed.
Capítulo 1 : Introdução à UML

Representação gráfica de classes (Diagrama de classes)

Perguntas?

Alexandre de Andrade Barbosa
alexandre.barbosa@arapiraca.ufal.br